POZNAN UNIVERSITY OF TECHNOLOGY

Wojciech Mioduszewski

Klasyfikacja danych opisanych za pomocą szeregów czasowych

Praca magisterska

Promotor: dr inż. Jerzy Błaszczyński

Poznań, 2015

Spis treści

Rozdział 1

Wstęp

Cel i zakres pracy

Cel: Opracowanie i implementacja różnych podejść do klasyfikacji danych czasowych.
Zadania:

- Zapoznać się z literaturą tematu.
- Opracować wybrane podejścia do klasyfikacji danych czasowych.
- Zaimplementować i udokumentować zaproponowan rozwiązania.
- Przeprowadzić eksperyment obliczeniowy

Początkowo celem niniejszej pracy była analiza szeregów czasowych zawierających dane ciśnienia w oku pacjentów zdrowych, oraz tych ze zdiagnozowaną jaskrą. Ponadto zamiarem było użycie do tego celu metody SAX, a następnie zbudowanie klasyfikatora potrafiącego sklasyfikować dane wytworzone przez tą metodę. Następnie, aby nie testować sposobów klasyfikacji tylko i wyłącznie na danych zebranych w celu oceny jaskry, lecz również sprawdzić jak wybrane i stworzone metody poradzą sobie w odniesieniu do innych szeregów czasowych, rozszerzono zbiór danych o kolejne zestawy szeregów czasowych. Kolejną rzeczą, od której należało się uniezależnić są klasyfikatory, dlatego też eksperymenty przeprowadzone zostały na kilku różnych technikach kategoryzowania instancji.

Ostatecznie po wielu testach wyklarowano sześć eksperymentów, których wyniki porównano i zinterpretowano. Składowe każdego z eksperymentów zostały dokładnie opisane.

W rozdziale 2 przedstawiono kilka podejść analizy szeregów czasowych wybranych z przeanalizowanej literatury. Ponadto zademonstrowano działanie dwóch kluczowych algorytmów tej pracy, to jest algorytmu SAX oraz DTW. Rozdział 3 poświęcono zobrazowaniu koncepcji projektu, toteż rozdział ten skupia się na przestudiowaniu różnych sposobów przetwarzania danych jeszcze zanim te trafią do klasyfikatora. Czytelnik znajdzie tu również krótki opis użytych klasyfikatorów, a także miary jakie użyto do oceniania dokładności klasyfikacji.

Rozdział 4 przedstawia sposób implemetacji programu potrzebnego do przeprowadzenia eksperymentów. Zamieszczono tutaj zwięzłe opisy użytych bibliotek, oraz szczegółowy opis importu oraz eksportu danych - tak aby w przyszłości nie było problemu z ewentualnym rozszerzeniem istniejącego kodu. Rozdział 5 zawiera analizę wyników oraz wnioski, a tym samym stanowi podsumowanie niniejszej pracy.

Rozdział 2

Kontekst teoretyczny

2.1 Definicja szeregu czasowego

Szereg czasowy jest to seria pewnych obserwacji osadzonych w czasie. Można powiedzieć, że jest to przyporządkowanie danych liczbowych do odpowiadających im punktów w czasie, najczęściej z jednakowymi odstępami między kolejnymi wartościami.

http://www.cs.put.poznan.pl/jstefanowski/aed/TPtimeseries.pdf

2.2 Przykładowe metody analizy danych czasowych[10]

Tak obszerny problem jak analiza danych czasowych musi nieść za sobą rozmaite metody przeprowadzania tej analizy. Pomimo tego, że do ostatecznego eksperymentu wybrano tylko jedną z nich - regresję, to postanowiono przedstawić po krótce czym charakteryzują się poszczególne metody oraz ewentualnie dlaczego zostały porzucone.

2.2.1 Regresja liniowa

Regresja w odniesieniu do danych czasowych sprowadza się do estymowania liniowego trendu jaki prezentuje badany szereg. Koncepcyjnie metoda polega na stworzeniu funkcji liniowej, która w najbardziej dokładny sposób przybliża wartości na kolejnych obserwacjach. Matematyczny model regresji ma zatem następującą postać:

$$y = ax + b$$

Jako miarę błędu przyjmuje się sumę kwadratów różnicy między oszacowaniami, a wartościami właściwymi.

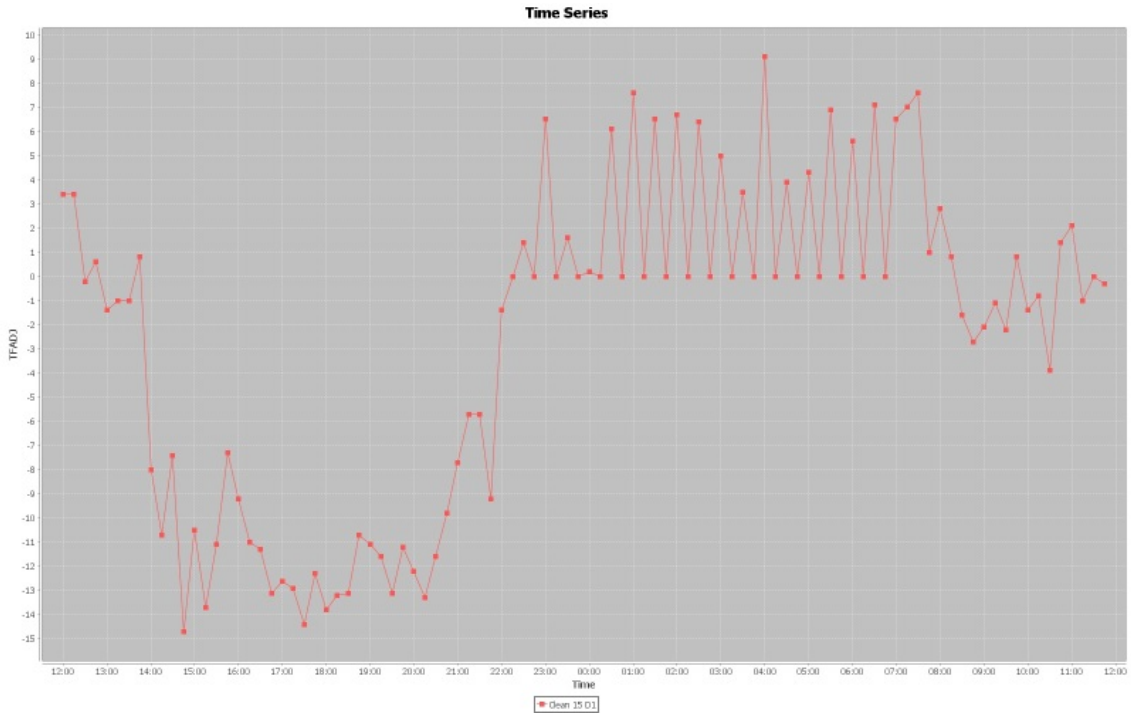
$$S = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

http://www.cs.put.poznan.pl/jstefanowski/aed/TPDregresjawieloraka.pdf

2.2.2 Wygładzanie (ang. smoothing)

Wygładzanie to metoda starająca się zniwelować ponadprzeciętne różnice wartości między kolejnymi pomiarami. Dzięki temu podejściu można dokładniej przyjrzeć się ogólnemu zarysowi funkcji, czy jej okresowymi trendami, kosztem precyzji. Podejście to pozwala zminimalizować ewentualne szumy z badanego zbioru. Metoda ta jako argument przyjmuje szerokość okna k, w ramach którego będą uśredniane wartości. W pierwszym kroku liczy się średnią z k pierwszych wartości szeregu, następnie przesuwa okno o jeden element i znów liczy średnią z k wartości. Wyjściowy zbiór dla n-elementowego zbioru będzie miał $n - k$ wartości.

Poniżej dla porównania zaprezentowano wykres miary TFADJ w przeciągu doby 2.1, oraz jego wygładzony odpowiednik 2.2. Jak widać operacja wygładzania zredukowała skrajne wychylenia szeregu i wyklarowała paraboliczną tendencję tego zbioru.



Rysunek 2.1: Czysty sygnał TFADJ

2.2.3 Modele ARIMA

Model ARIMA (ang. Autoregressive Integrated Moving Average) koncepcyjnie składają się z trzech części - jest to autoregresja, integracja oraz średnia ruchoma.

Autoregresja jest to idea, która skupia się na wyrażeniu bieżącej wartości na podstawie$n$ poprzednich. Dla przykładu, funkcja dla $n = 2$ wygląda w ten sposób:

$$x_t = \omega_{t-2}x_{t-2} - \omega_{t-1}x_{t-1} + \omega_t$$

Średnia ruchoma ideowo jest bardzo zbliżona do autoregresji, jednak skupia się na zaburzeniach (ang. lags) w szeregu, a nie bezpośrednio na wartościach. Wzór przedstawia

Rysunek 2.2: Wygładzony sygnał TFADJ

się analogicznie jak w autoregresji.

Integracja z kolei pozwala na zastosowanie modelu ARIMA do procesów niestacjonarnych, które da się sprowadzić do procesów stacjonarnych dzięki przekształceniu oryginalnego sygnału na różnice (ang. difference equations) pomiędzy wartością obecną, a poprzednią.

Podsumowując, modele ARIMA doskonale spisują się w prognozowaniu wartości, bazując na danych historycznych szeregu, jednak nie są najlepszym wyborem jako metoda klasyfikacji. Ponadto metoda nie jest najłatwiejsza matematycznie. Biorąc pod uwagę te argumenty, metoda ta została odrzucona.

2.2.4 Analiza spektralna (widmowa)

Celem analizy spektralnej (ang. spectral analysis) szeregów czasowych jest zidentyfikowanie najczęściej powtarzających się wzorców w czasie, a następnie przybliżenie danego szeregu do procesu okresowego określonego wzorem:

$$x_t = U_1 \cos(2\Pi\omega t) + U_2 \sin(2\Pi\omega t)$$

gdzie amplituda funkcji to $A = \sqrt{U_1^2 + U_2^2}$.

Jak widać jest to kolejn metoda, za którą stoi solidne matematyczne zaplecze i która wymierzona jest w szeregi cykliczne, z którymi w zbiorze danych pacjentów jaskry nie mamy do czynienia, zatem metodę tę odrzucono.

2.3 SAX (Symbolic Aggregate approXimation)

Już na początkowym etapie rozważań nad zakresem niniejszej pracy zadecydowano, że jednym głównych nurtów jaki przyjmie, będzie przekształcanie sygnału według metody SAX. Mówiąc po krótce, jest to metoda odzwierciedlająca oryginalny sygnał w ciąg znaków, co stwarza możliwości klastfikacji danych za pomocą istniejących już algorytmów tekstowych.

2.3.1 Założenia

Korzystając z pomysłu prof. Eamonna przyjęto założenie, że dla szeregów czasowych o równych odstępach czasowych między pomiarami, można pominąć dane dotyczące momentu, w którym pomiaru dokonano i uprościć ten szereg do sekwencji samych wartości. W następujących rozdziałach przyjęto że operuje się na zbiorze N-elementowym o postaci $x_1, x_2, \ldots, x_n$

2.3.2 Schemat działania

Stworzenie reprezentacji SAX dla szeregu czasowego składa się z trzech etapów:

2.3.2.1 Z-normalizacja szeregu czasowego

Na potrzeby przedstawienia wzoru dla odchylenia standardowego przyjęto następującą definicję sumy elementów zbioru:

$$s_j = \sum_{k=1}^{N} x_k^j$$

dzięki któremu można przedstawić wzór na odchylenie standardowe tak jak poniżej[2]:

$$std = \sqrt{\frac{Ns_2 - s_1^2}{N(N-1)}}$$

Mając obliczone odchylenie standardowe i średnią wartość dla zbioru, którą wyrażono poniżej symbolem $mean$ można przejść do właściwej normalizacji, co oznacza przekształcenie każdego z elementów zbioru w sposób przedstawiony poniżej:

$$x_i = \frac{x_i - mean}{std}$$

2.3.2.2 Dyskretyzacja sygnału za pomocą algorytmu PAA (Piecewise Aggregate Approximation)

Jednym z argumentów, jaki przyjmuje algorytm SAX jest długość łańcucha wyjściowego - co w praktyce sprowadza się do tego na ile części podzielić długość szeregu czasowego. Oznacza to, że metoda zmniejsza wymiarowość szeregu z N do żądanych M elementów. Przebieg algorytmu jest intuicyjny - sekwencja $x_1, x_2, \ldots, x_N$transformowana jest do postaci $y_1 y_2, \ldots, y_M$ poprzez podział sekwencji N-elementowej na M równych części według

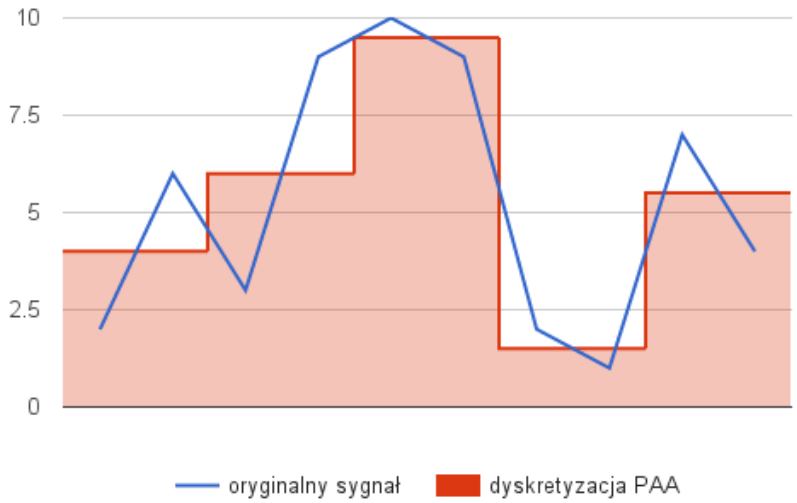
wzoru[8]:

$$y_i = \frac{M}{N} \sum_{j=\frac{N}{M}(i-1)+1}^{(N/M)i}$$

Poniżej przedstawiono przykładową dyskretyzację dla następującego szeregu czasowego:

i	1	2	3	4	5	6	7	8	9	10
oryginalny sygnał x_i	2	6	3	9	10	9	2	1	7	4

dzieląc szereg dziesięcioelementowy na pięć równych części algorytmem PAA[1]: Dzięki



Rysunek 2.3: Porównanie oryginalnego szeregu czasowego i jego 5-elementowej dyskretyzacji

temu otrzymano redukcję pięciowymiarową:

i	1	2	3	4	5
zdyskretyzowany sygnał y_i	4	6	9.5	1.5	5.5

2.3.2.3 Algorytm SAX

Jak wspomniano wyżej, jednym z argumentów jakie przyjmuje algorytm SAX jest długość sekwencji wyjściowej, zaś drugim - liczba dostępnych liter w alfabecie. Ta liczba precyzuje na ile części podzielić zbiór wartości danego (przetworzonego już) szeregu, a tym samym jakie indeksy (znaki) przypisać wartościom lądującym w poszczególnych zakresach. Do przypisania znaków dla odpowiednich przedziałów potrzebny jest przedział danych zatem obliczyć różnicę między elementem minimalnym a maksymalnym zbioru y

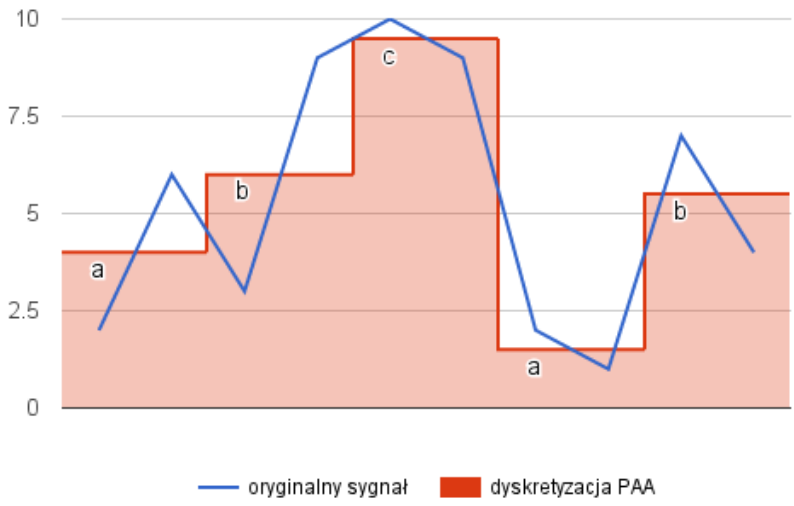
$$d = |max(y) - min(y)|$$

a następnie ustalić krok k, ustalający rozmiar okna, dzieląc zbiór wartości d przez liczbę części M. W ostatnim kroku, począwszy od wartości minimalnej przydzielamy kolejne przedziały kolejnym znakom. Dla powyższego przykładu przyjęto alfabet trójelementowy

$alpha = a, b, c$. Dla $d = 9.5 - 15 = 8$ wielkość okna wynosi $k = \frac{8}{3} = 2\frac{2}{3}$. Co prowadzi do podziału:

a	b	c
$< 1\frac{1}{2}, 4\frac{1}{6})$	$< 4\frac{1}{6}, 6\frac{5}{6})$	$< 6\frac{5}{6}, 9\frac{1}{2} >$

dzięki czemu otrzymujemy przypisania naniesione na wykres: co oznacza, że ostatecz-



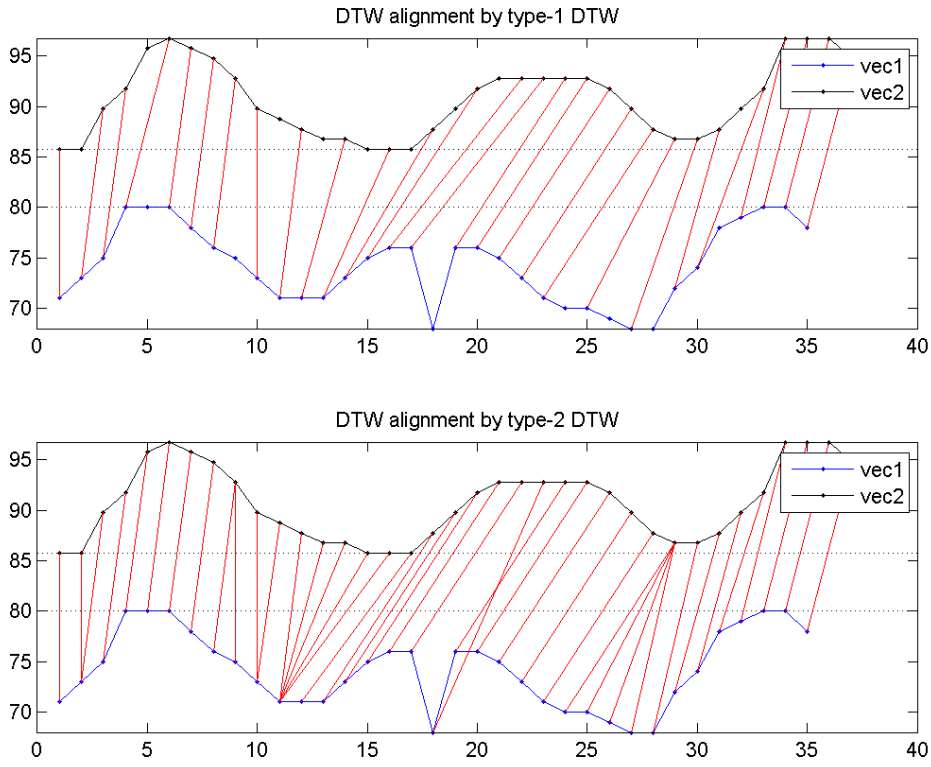
Rysunek 2.4: Oryginalny sygnał z dyskretyzacją i naniesionym SAXem

nym wynikiem metody SAX dla szeregu czasowego x przy zadanych parametrach długości łańcucha wyjściowego 5 oraz alfabetu trójelementowego jest ciąg znaków *abcab*.

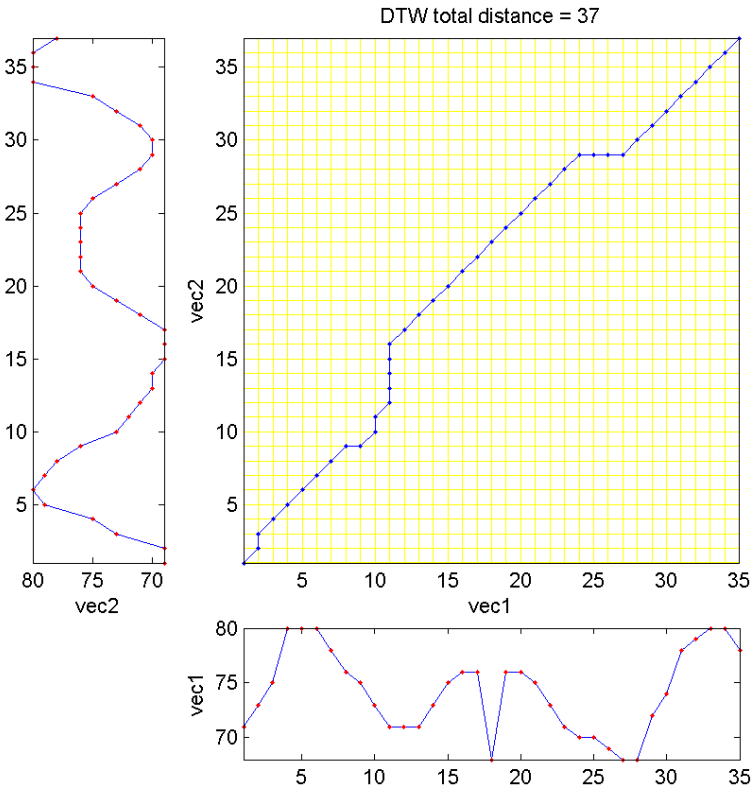
2.4 DTW (Dynamic Time Warping)

Algorytm DTW jest metodą, która stara się wyeliminować błąd porównywania dwóch szeregów czasowych, których charakterystyczne cechy były wobec siebie nieco przesunięte w czasie bądź w jednym szeregu zdarzenie postępowało wolniej niż w drugim - choć nadal było to to samo zdarzenie. Potrafi też poradzić sobie z pojedynczymi błędami pomiaru czy drobnymi szumami. Te cechy widać na rysunku 2.5 na którym sygnał `vec1` wyraźnie dostosowywuje się do sygnału `vec2` mimo dużego spadku wartości w połowie przebiegu.

To jak realizowany jest algorytm DTW obrazuje rysunek 2.6. W momencie przyrównywania dwóch sygnałów, tworzona jest macierz odległości, której wartiści obliczane są rekurencyjnie. Ostatni wiersz macierzy to różnice między kolejnymi wartościami szeregów. Kolejne wiersze powstają na ich podstawie. Gdy maciesz jest już skonstruowana, wybiera się najkrótszą kosztowo ścieżkę między skrajnymi wierzchołkami macierzy - suma wartości komórek, przez które przechodzi ścieżka jest dystansem między tymi szeregami. Należy nadmienić, że metoda ta jest dokładna, lecz za jej skutecznością idzie spory nakład czasowy - czas przetwarzania jest nieporównywalnie większy od pozostałych metod. Szczegółowo zostanie to omówione w rodziale poświęconym wynikom.



Rysunek 2.5: Dopasowywanie DTW[7]



Rysunek 2.6: Macierz DTW[7]

Teoretycznie metoda jako parametr przyjmuje szerokość okna DTW - określa to ramy części macierzy, na której można szukać ścieżki pomiędzy wierzchołkami. Jednak ze względu na wspomniany już ogromny narzut czasowy, metoda została odpalona tylko na jednej konfiguracji o najmniejszym możliwym - dwuelementowym - oknie DTW.

2.5 Badane zbiory danych

Jak wspomniano wcześniej, podstawowy zbiór, który miał zostać zbadany to pacjenci z jaskrą (lub bez). Z tego powodu będzie on opisany nieco szerzej niż pozostałe zbiory

2.5.1 Pacjenci

Zbiór dostarczony przez dr. J. Błaszczyńskiego. Zbiór zawiera pomiary dla 116 pacjentów, z których 65 to pacjenci zdrowi, a 51 to pacjenci ze zdiagnozowaną jaskrą. Dla każdego pacjenta zebrano 288 pomiarów (pomiar co 5 minut przez całą dobę) tzw. TFADJ <tu rozwinięcie?> będącym przekształceniem zmierzonego w danym momencie ciśnienia w oku. Poniżej poglądowe zdjęcie metody zbierania pomiarów:

Rysunek 2.7: Metoda zbierania pomiarów od pacjentów

2.5.2 Dane prof. Eamonna Keogh'a[5]

Zbiory pobrane od prof. Eamonn'a nie zawierają opisu poszczególnych zbiorów, dlatego też ich tu nie przedstawiono. Jednak przetwarzając dane można było wyciągnąć z nich pewne interesujące statystyki, które przedstawiono w poniższej tabeli (w ostatnim wierszu porównawczo zestawiono zbiór dr. Błaszczyńskiego):

Tablica 2.1: Szczegóły danych

nazwa zbióru	**liczba instancji**	**liczba klas**	**podział na klasy**	**długość szeregu**
ECG200	200	2	'-1' - 67 rekordów '1' - 133 rekordów	96
ECGFiveDays	884	2	'1' - 442 rekordów '2' - 442 rekordów	136
TwoLeadECG	1162	2	'2' - 581 rekordów '1' - 581 rekordów	82
Yoga	3300	2	'1' - 1530 rekordów '2' - 1770 rekordów	426
MoteStrain	1272	2	'2' - 587 rekordów '1' - 685 rekordów	84
ItalyPowerDemand	1096	2	'1' - 547 rekordów '2' - 549 rekordów	24
ChlorineConcentration	4307	3	'1' - 1000 rekordów '3' - 2307 rekordów '2' - 1000 rekordów	166
Two Patterns	5000	3	'2' - 1248 rekordów '3' - 1245 rekordów '4' - 1201 rekordów '1' - 1306 rekordów	128
Wafer	7174	2	'1' - 6402 rekordów '-1' - 762 rekordów	152
InlineSkate	650	7	'2' - 100 rekordów '3' - 103 rekordów '7' - 62 rekordów '6' - 98 rekordów '4' - 108 rekordów '5' - 117 rekordów '1' - 62 rekordów	1882
Pacjenci	*116*	*2*	*'0' - 65 rekordów* *'1' - 51 rekordów*	*288*

Rozdział 3

Koncepcja projektu

3.1 Metody przetwarzania danych dla klasyfikatorów

W celu osiągnięcia jak najlepszych wyników, postanowiono przy rozpoczęciu prac nad tą pracą dyplomową spróbować podejścia polegającego na zmodyfikowaniu sygnału wejściowego przed przekazaniem go do klasyfikatora. Zrealizowano to na kilka sposobów, które zostały opisane w niniejszym podrozdziale.

3.1.1 Dzielenie sygnału na części

Pierwszym oraz jeden z prostszych koncepcyjnie pomysłów jaki został wdrożony było podzielenie wejściowego sygnału na części. Przykładowo dla regresji znacznie dzięki temu wzrasta liczba atrybutów, a co za tym idzie - dokładność klasyfikacji. Tak jak klasyczna regresja zwróci dwa argumenty - parametr `a` oraz `b`, tak po podzieleniu sygnału na 3 części dostaniemy aż trzy pary takich parametrów. Jako możliwe wartości tego parametru na przestrzeni eksperymentów wybrano 1, 3 oraz 5. Dodatkowo dla eksperymentu z regresją dodano dodatkowe 10, 20 oraz 48.

3.1.2 Podejście SAX

Opisana wyżej metoda dyskretyzacji sygnału została użyta do kilku spośród przeprowadzonych eksperymentów. Żeby zbadać różne możliwości przebiegu tej dyskretyzacji zdecydowano się na parametryzację tej metody. W ramach dopuszczalnych długości alfabetu wybrano 3, 5 oraz 11 znaków, a w kwestii długości łańcucha wynikowego zdecydowano się na stałą długość 96 znaków.

3.1.3 Zliczanie (eksperyment „Counted")

Metoda zliczania ściśle bazuje na metodzie SAX - modyfikacje danych następują na poziomie wynikowego łańcucha znaków, a nie czystego szeregu czasowego. Pierwszym krokiem zatem jest wyliczenie łańcucha SAX dla instancji. Następnie łancuch ten dzieli się na zadaną liczbę części. W kolejnym kroku potrzebny jest kolejny parametr - ngram - który może przyjąć wartość 2 lub 3. Polega on na zliczeniu wszystkich możliwych 2- lub 3gra-

mów w łańcuchu badanej części. Wynikiem są wystąpienia danych ngramów w każdej z części. Poniżej przedstawiono to na przykładzie.

Dla tego przykładu przyjęto podane założenia: metoda otrzymuje 3 rekordy, a dla ich sygnałów metoda SAX wyliczyła odpowiednie łańcuchy:

- r1: 'abaabbbbaabb' z przypisaniem do klasy '1'
- r2: 'abbabaaaabba' z przypisaniem do klasy '1'
- r3: 'aaababbbaaaa' z przypisaniem do klasy '2'

W tabeli zamieszczono przebieg algorytmu dla podziału na 3 części oraz wyszczególniono 2gramy pojawiające się w ramach każdej z części.

	okres 1	okres 2	okres 3
rekord 1	**abaa**	**bbbb**	**aabb**
	ab	bb	aa
	ba		ab
	aa		bb
rekord 2	**abba**	**baaa**	**abba**
	ab	ba	ab
	bb	aa	bb
	ba		ba
rekord 3	**aaab**	**abbb**	**aaaa**
	aa	ab	aa
	ab	bb	

Tablica 3.1: Podział łańcuchów SAX instancji na 2gramy 'Counted'

W kolejnym kroku należy zliczyć ile razy wystąpił każdy z 2gramów. Wyniki przedstawia tabela

	rekord 1	rekord 2	rekord 3
o1ab	1	1	1
o1ba	1	1	0
o1aa	1	0	2
o1bb	0	1	0
o2bb	3	0	2
o2ba	0	1	0
o2aa	0	2	0
o2ab	0	0	1
o3aa	1	0	3
o3ab	1	1	0
o3bb	1	1	0
o3ba	0	1	0

Tablica 3.2: Zliczenie 2gramów dla instacji przedstawionych w 3.1

Na podstawie tabeli 3.2 generowany jest plik wejściowy arffdla klasyfikatora, który dla omawianego przykładu wyglądałby tak, jak pokazano w listingu 3.1.

Listing 3.1: Wynikowy plik arff dla eksperymentu 'Counted'

```
@relation 'Sax counted'

@attribute o1ab numeric
@attribute o1ba numeric
@attribute o1aa numeric
@attribute o1bb numeric
@attribute o2bb numeric
@attribute o2ba numeric
@attribute o2aa numeric
@attribute o2ab numeric
@attribute o3aa numeric
@attribute o3ab numeric
@attribute o3bb numeric
@attribute o3ba numeric

@attribute destClass {1,2}

@data
1,1,1,0,3,0,0,0,1,1,1,0,1
1,1,0,1,0,1,2,0,0,1,1,1,1
1,0,2,0,2,0,0,1,3,0,0,0,2
```

3.1.4 Dominacja (eksperyment „Dominant")

Eksperyment nazwany eksperymentem dominacyjnym jest łudząco podobny do eksperymentu 'Counted' - różni się jedynie sposobem zliczania ngramów, dlatego zostanie zobrazowany na tym samym przykładzie. Tym razem analizując każdy z ngramów w poszczególnej części zlicza się ngramy będące co najmniej tak samo „wysokie" na każdym z argumentów (tutaj: znaków w łańcuchu) w opcji 'atMost' lub też co najwyżej tak samo opisany na każdym z agrumentów ('atLeast') jak ten badany. W tabeli 3.3 przedstawiono jak wyglądają wyniki zliczania z dominacją dla rekordów przedstawionych w tabeli 3.1.

	rekord 1	rekord 2	rekord 3
o1<=ab	2	1	3
o1>=ab	1	2	1
o1<=bb	3	3	3
o1>=bb	0	1	0
o1<=ba	2	1	2
o1>=ba	1	2	0
o1<=aa	1	0	2
o1>=aa	3	3	3

Tablica 3.3: Dominacja ngramów dla rekordów z tabeli 3.1

Jak można zauważyć, w takim podejściu podwaja się liczba argumentów dla każdego rekordu (w tabeli przestawiono tylko część dotyczącą pierwszego okresu - stąd prefiksy 'o1'). Wartość '2' dla rekordu 1 w wierszu o1<=ab oznacza, że spośród ngramów w tym okresie dwa z nich były co najwyżej takie jak 2gram 'ab' - są to 2gramy 'ab' oraz 'aa' - oba występowały jednokrotnie. Z kolei na przykład wartość 0 dla rekordu 3 w wierszu o1>=ba oznajmia, że nie było żadnego 2gramu większego lub równego 2gramowi 'ba' w łańcuchu odzwierciedlającym przebieg sygnału dla rekordu nr 3.

Jak zauważono, dla ngramów prezentujących co najmniej najmniejsze lub co najwyżej największe wartości na każdym ze znaków łańcucha, wartości zliczeń są zawsze maksymalne (w tym przykładzie wiersze 'o1<=bb' oraz 'o1>=aa'). Nie wnoszą zatem one żadnej informacji godnej uwagi klasyfikatora, zatem wiersze te były wycinane przed eksportem do pliku arff.

Ze względów technicznych w plikach arff znaki '<=', '>=' zostały zmienione na 'atL', 'atM' czyli odpowiednio atLeast (co najmniej) oraz at Most (co najwyżej). W listingu 3.2 przedstawiono pełny plik arff dla omawianego przypadku w wariancie eksperymentu 'Dominant'.

Listing 3.2: Wynikowy plik arff dla eksperymentu 'Dominant'

```
@relation 'Sax dominant'

@attribute o1atMaa numeric
@attribute o1atMab numeric
@attribute o1atLab numeric
@attribute o1atMba numeric
@attribute o1atLba numeric
@attribute o1atLbb numeric
@attribute o2atLbb numeric
@attribute o2atMaa numeric
@attribute o2atMba numeric
@attribute o2atLba numeric
@attribute o2atMab numeric
@attribute o2atLab numeric
@attribute o3atMaa numeric
@attribute o3atLbb numeric
@attribute o3atMab numeric
@attribute o3atLab numeric
@attribute o3atMba numeric
@attribute o3atLba numeric
@attribute destClass {1,2}

@data
1,2,1,2,1,0,3,0,0,3,0,3,1,1,2,2,1,1,1
0,1,2,1,2,1,0,2,3,1,2,0,0,1,1,2,1,2,1
2,3,1,2,0,0,2,0,0,2,1,3,3,0,3,0,3,0,2
```

3.1.5 K najbliższych sąsiadów (ang. KNN - k Nearest Neighbors)

W eksperymencie zorientowanym wokół algorytmu KNN zdecydowano się na dość nowatorskie podejście z dyskretyzacją danych. Przeważnie używając algorytmu najbliższych sąsiadów, za dane wejściowe uznaje się czysty zebrany sygnał. W tym projekcie postanowiono zdyskretyzować dane używając do tego metody SAX, a dopiero potem przekazać jej rezultat do klasyfikatora KNN, dzięki czemu zyskaliśmy swego rodzaju normalizację danych.

3.1.6 Bagging

Przygotowanie danych dla eksperymentu Bagging przebiegało dokładnie w ten sam sposób co dla eksperymentu Zliczającego. Eksperymenty różniły się sposobem użycia klasyfikatora. Oba eksperymenty skorzystały z `J48`, lecz dla Baggingu był on opakowany w meta-klasyfikator `weka.classifiers.meta.Bagging`, który bliżej zostanie opisany w podrozdziale (3.2.5).

3.2 Użyte klasyfikatory

3.2.1 J48

J48 to jeden z podstawowych klasyfikatorów pakietu `weka`, opierający algorytm klasyfikacji na drzewach decyzyjnych. Jego działanie rozpoczyna się na przypisaniu wszystkich instancji treningowych do jednego liścia. Jeżeli wszystkie elementy należą do tej samej klasy, to algorytm się kończy - jeśli nie, wybiera atrybut, który najlepiej rozdziela instancje w liściu na różne klasy i tworzy odpowiedni liść. Algorytm trwa aż w liściach pozostaną jednorodne grupy elementów lub nie będzie można dobrać kolejnych atrybutów jako kryterium podziału.

3.2.2 IBk

Klasyfikator stworzony do przeprowadzania klasyfikacji za pomocą algorytmu k-najbliższych sąsiadów (ang. KNN: K-Nearest-Neighbors) z pakietu `weka`. Dzięki temu, że klasyfikator ten można w łatwy sposób zmodyfikować na poziomie kodu, dostosowano go również do eksperymentu DTW, który również bazuje na metodzie KNN, jednak używa innego algorytmu wyszukiwania ów najbliższych sąsiadów.

3.2.3 VCDomLem[3]

VCDomLem

3.2.4 NgramClassifier

Autorski klasyfikator zbudowany na wzór klasyfikatora pakietu weka (rozszerza klasę `weka.classifiers.Classifier`). Jako wejście przyjmuje jeden argument - łańcuch wyjściowy z algorytmu SAX. Następnie dla każdej klasy decyzyjnej wyznacza liczności występowań danych n-gramów (gdzie n jest parametrem metody). W momencie klasyfikacji instancji zlicza się występowanie ngramów w ramach badanej instancji, by potem zmierzyć dystanse dzielące instancję oraz wszystkie klasy decyzyjne. Instancja zostaje przypisana do klasy, z którą dzieli ją najmniejszy dystans. Aby przedstawić sposób liczenia tej odległości zdefiniowano następującą funkcję: $occ(klasa, ngram)$ która jako wynik zwraca ile wystąpień danego ngramu miało miejsce w ramach podanej klasy decyzyjnej. Analogicznie $occ(instancja, ngram)$ podaje liczbę wystąpień ngramu dla zadanej instancji.

Zatem dla każdej klasy liczony jest następująca odległość między instancją a klasą:

$$dist(instance, class) = \sqrt{\sum_{ngram}^{instance} (occ(klasa, ngram) - occ(instancja, ngram))^2}$$

3.2.5 Bagging

Attribute Bagging[4] to

3.3 Miary[6]

Do oceniania poprawności eksperymentów wybrano sześć miar statystycznych. Niektóre z nich bazują na kilku definicjach c *confussion matrix*, których znaczenie przedstawiono w tabeli poniżej:

	Rzeczywista wartość pozytywna	Rzeczywista wartość negatywna
Przewidywana wartość pozytywna	**True Positive**	**False Positive**
Przewidywana wartość negatywna	**False Negative**	**True Negative**

3.3.1 Dokładność (ang. accuracy)

Najbardziej znana i intuicyjna miara ewaluacji klasyfikatora to właśnie dokładność. Określa ją stosunek poprawnie sklasyfikowanych instacji do liczby wszystkich instacji. Dzięki swojej prostej koncepcji może być z powodzeniem używana zarówno w problemach binarnych jak i wieloklasowych.

$$Accuracy = \frac{correctly\,classified\,instances}{total\,instances}$$

3.3.2 Wrażliwość (ang. sensitivity)

Miara ta, znana również pod nazwą *True positive rate*, skupia się na zobrazowaniu stosunku instacji poprawnie zidentyfikowanych w klasie umownie uznanej za pozytywną do liczby instacji w tej klasie. Na przykładzie zbioru Pacjentów opisywanego w tej pracy można powiedzieć, że jest to proporcja pacjentów poprawnie sklasyfikowanych jako chorzy na jaskrę do wszystkich, którzy są na tę jaskrę chorzy. Wzór na wrażliwość podano poniżej:

$$Sensitivity = \frac{TP}{TP + FN}$$

3.3.3 Specyficzność (ang. specificity)

Specyficzność to w pewnym sensie przeciwieństwo wrażliwości - uwydatnia proporcję instancji dobrze sklasyfikowanych w klasie uznanej umownie za negatywną do liczby instancji w tej klasie. Ponownie odwołując się do analizowanych tutaj Pacjentów wynik specyficzności to iloraz pacjentów prawidłowo sklasyfikowanych jako zdrowi i liczby wszystkich zdrowych pacjentów. Poniżej przedstawiono jej wzór ogólny:

$$Specificity = \frac{TN}{TN + FP}$$

3.3.4 Średnia geometryczna (ang. geometric mean)

Miara *G-mean* to geometryczne uśrednienie miar wspomnianych powyżej, czyli wrażliwości i specyficzności. Estymuje więc skuteczność klasyfikatora zarówno na klasie pozytywnej jak i negatywnej. Dla formalności poniżej przedstawiono jej wzór:

$$Gmean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} = \sqrt{Sensitivity \times Specificity}$$

Jak można zauważyć dalej w wynikach, dla zbiorów wieloklasowych miary Sensitivity, Specificity oraz GMean nie były obliczane, ze względu na to, że musiałyby zostać przedstawione dla każdej klasy osobno. Porównywanie takich wyników między zbiorem trójklasowym a siedmioklasowym byłoby niemiarodajne, dlategoteż w zbiorach posiadających co najmniej 3 klasy zrezygnowano z obliczania tych trzech miar.

3.3.4.1 Miara F (ang. F-measure, F1)

Na potrzeby zobrazowania miary F posłużono się definicjami precyzji (ang. precision) oraz przywołania (ang. recall). Precyzja wyraża to ile było trafień w klasie pozytywnej w odniesieniu do wszystkich instancji, które klasyfikator uznał za należące do klasy pozytywnej ($Precision = \frac{TP}{TP+FP}$). Przywołanie natomiast to nic innego co wspomniana już wcześniej wrażliwość. Miara F1 zaś jest dwukrotnym ilorazem iloczynu precyzji i przywołania oraz ich sumy:

$$F1 = 2\frac{precision \times recall}{precision + recall}$$

3.3.5 Procent poprawnie sklasyfikowanych (ang. Percent Correctly Classified)

Jest to prosta miara uwidaczniająca udział prawidłowo sklasyfikowanych instancji w odniesieniu do ogółu. Jej wzór prezentuje się następująco:

$$PCC = \frac{correctly\,classified\,instances}{instances}$$

3.4 Posumowanie

Jak szczegółowo opisano wyżej, przeprowadzono aż 7 zróżnicowanych eksperymentów na 11 zbiorach danych. W tabeli 3.4 przedstawiono zebrane informacje dotyczące tego jak przetworzono dane zanim przekazano je do klasyfikatora oraz jaki klasyfikator te dane przeliczał. Dla przejrzystości użyto skrótu 'sax' dla pary parametrów: rozmiar ngramu oraz liczba liter w alfabecie.

Nazwa eksperymentu	**dane wejściowe**	**klasyfikator**	**parametry**
Regresja	oryginalny sygnał	J48	części
Dominacja	zliczone dominujące n-gramy SAX	VCDomLem	części, sax
Zliczanie	zliczone n-gramy SAX	J48	części, sax
Ngram	SAX	NgramClassifier	sax
KNN	sygnał zdyskretyzowany metodą SAX	IBk	sax, k
DTW	oryginalny sygnał	IBk	k, okno dtw
Bagging	zliczone n-gramy SAX	J48	części, sax

Tablica 3.4: Zrealizowane eksperymenty

Rozdział 4

Implementacja

4.1 Koncepcja budowy biblioteki

Wiodącym założeniem przyświecającym pracom implementacyjnym nad biblioteką obliczeniową było podążanie za dobrymi praktykami wytwarzania oprogramowania według Roberta C. Martina - z głównym naciskiem na OCP (Open Closed Principle)[9] - zasadę „otwarte-zamknięte". Oznacza to, że oprogramowanie ma być otwarte na zmiany, lecz zamknięte na modyfikacje - starano się zatem osiągnąć stan, w którym istniejący kod można rozszerzyć z jak najmniejszym wysiłkiem wniesionym w edycję istniejącego kodu. Szczegóły tej idei przedstawiono w poniższych sekcjach.

4.1.1 Użyte biblioteki pomocnicze

Żeby móc w pełni opisać funkcjonalność i sposób budowy programu należy nadmienić jakimi bibliotekami wspomagano się podczas tego procesu.

4.1.1.1 Weka

Najważniejsza biblioteka, z którą program współpracuje najściślej. Wiedząc, że Weka jest często używana do klasyfikacji czy uczenia maszynowego, autor chciał stworzyć narzędzie kompatybilne z programem stworzonym na Uniwesytecie w australijskim Waikato. Kilka kluczowych klas programu autora rozszerza klasy pakietu `weka` stwarzając możliwości wykorzystania rozmaitych składowych tego systemu. Ponadto używanie klas takich jak `weka.classifiers.Evaluation` do przeprowadzania eksperymentów daje pewność, że wykorzystywane rozwiązanie jest dobrze przetestowane i powinno zwracać rzetelne wyniki. Stąd również przyjęto format wyjściowy plików klasyfikacyjnych `*.arff`

4.1.1.2 jMotif

Z pakietu `jMotif` zaczerpnięto metody zajmujące się tworzeniem łańcucha danych na podstawie zadanego szeregu czasowego za pomocą algorytmu SAX.

4.1.1.3 JFreeChart

Biblioteka służąca do rysowania wykresów. Bardzo pomocna w początkowej fazie implementacji, kiedy autorowi zależało na zobrazowaniu przebiegu szeregów szasowych. Ich wizualizacja mogła przyczynić się do lepszego zrozumienia zależności między przebiegami, a diagnozą pacjenta.

4.1.1.4 Jxl

Dzięki pakietowi `jxl`w początkowej fazie implementacji autor mógł ściśle współpracować z arkuszami kalkulacyjnymi w celu raportowania danych do plików .xls w celu ich dalszej analizy statystycznej.

4.1.1.5 joda-time

W momencie gdy prace nad tą pracą dyplomową zostały rozpoczęte, wspracie dla obiektów typu data czy czas było dla autora niewystarczające, dlatego zdecydowano się na wykorzystanie biblioteki `joda-time` w celu zapewnienia wymaganej obsługi tego rodzaju danych.

4.1.1.6 fast-dtw

Pakiet fast-dtw dostarcza obsługę obliczeniową dla eksperymentu bazującego na algorytmie Dynamic Time Warping, dzięki czemu autor mógł skorzystać z gotowego, przetestowanego i relatywnie wydajnego czasowo i pamięciowo rozwiązania dla tego celu.

4.1.1.7 apache-commons

W momentach gdy brakowało pewnych struktur danych w obecnej wersji Java - takich jak choćby Pair (ang. para) lub należało skorzystać z popularnych metod takich jak obliczenie regresji, korzystano z bibliotek apache-commons.

4.1.2 Punkt wyjścia

Praca programu zorientowana jest wokół obiektu `Workflow` (ang. „przepływ”, „proces”) - ma on na celu zdefiniowanie następujących po sobie kroków składających się na przebieg eksperymentu - począwszy od wczytania danych, aż do wypisania ostatecznego wyniku. Jego rdzeń zawiera się w abstrakcyjnej klasie `WorkflowBase`która posiada trzy główne metody - odpalenia przebiegu pełnego eksperymentu, wygenerowania pliku arff dla eksperymentu oraz wyliczenia wyniku eksperymentu dla zadanego pliku arff. Ponadto jest w stanie opisać podstawowe statystyki wczytanych danych, takie jak liczba klas decyzyjnych, podział instacji w klasach, czy liczba atrybutów w ramach instancji. Aby stworzyć własny eksperyment wystarczy stworzyć klasę dziedziczącą z `WorkflowBase` oraz zaimplementować wymagane metody. Przykładowy szkic takiej klasy przedstawiono na listingu 4.1: Jeżeli wczytane przez nas dane wymagają jakiejkolwiek ingerencji obliczeniowej - tak jak na przykład eksperyment Counted, to fragment kodu odpowiadający za te obliczenia powinien znaleźć się w metodzie `processData`. Jak widać w przywołanym listingu, wszystkie

Listing 4.1: Kod eksperymentu Counted

```
public class CountedWorkflow extends WorkflowBase {

    private List<CalculatedRecord> calculatedRecords;

    public CountedWorkflow(DivisionOptions divisionOption, boolean glaucoma) {
        super(divisionOption, glaucoma);
    }

    @Override
    protected Instances buildInstances() {
        exporter = new CountedSaxArffBuilder(calculatedRecords);
        return exporter.buildInstances();
    }

    @Override
    protected void processData() throws Exception {
        calculatedRecords = new ArrayList<CalculatedRecord>();

        for (IRecord record : records) {
            ArrayList<HashMap<String, Integer>> periodicallyCountedNgrams =
                                    new ArrayList<HashMap<String, Integer>>();

            List<String> dividedSax = DataDivider.divideStringRegularly(
                    record.getSaxString(), divisionPartsAmount);

            for (String elem : dividedSax) {
                HashMap<String, Integer> ngramCountMap = PeriodicNgramCounter
                        .slashStringAndCountNgrams(elem, windowLen);
                periodicallyCountedNgrams.add(ngramCountMap);
            }

            CalculatedRecord calcRecord = new CalculatedRecord(
                    periodicallyCountedNgrams, record.getDestinationClass());
            calculatedRecords.add(calcRecord);
        }
    }

    @Override
    protected void setConcerningParams() {
        concerningParameters.add(Pair.of("parts", divisionPartsAmount));
        concerningParameters.add(Pair.of("ngram", windowLen));
        concerningParameters.add(Pair.of("alpha", Config.getInstance().getSaxAlphabeatSize()));
    }
}
```

wymagane kalkulacje odbywają się w tym fragmencie kodu. Kolejna metoda to `buildInstances` która odpowiada za ustawienie odpowiedniego eksportera danych (w szczegółach zostanie omówiony niżej) oraz przekazanie mu przeliczonych danych. Za jego pomocą tworzony jest obiekt `Instaces`, który musi zostać zwrócony jako wynik metody. Funkcja `setConcerningParams`służy już tylko do tego, aby podczas wyświetlania wyniku eksperymentu wypisać, które parametry programu mają wpływ na pracę tego eksperymentu, oraz jakie ich wartości zostały ustawione podczas bierzącego wykonania.

4.1.3 Importowanie danych

Importowanie danych zostało rozdzielone ze względu na dwa różne formaty danych jakie potrafi wczytać program. Ostateczną wersją danych pacjentów jest opisanie danych każdego pacjenta w osobnym pliku `.csv`. W każdym z nich podany jest identyfikator pacjenta, oraz dane z poszczególnych pomiarów ciśnienia w oku. W ramach każdego pomiaru udostępnionych jest sporo danych, takich jak godzina o której pomiar został zdjęty, pozycja ciała w której znajdował się pacjent itp. choć na potrzebę tej pracy korzystano tylko z miary `TFADJ`. Jedyna informacja, której nie ma w tym pliku to diagnoza - ta opisana jest w pliku zbiorczym dla wszystkich pacjentów. W kolejnych wierszach znajdują się pary identyfikator pacjenta - diagnoza.

Drugim formatem, jaki wspiera program jest format danych prof. Eamonna. Ich charakterystyka jest prosta - każdy nowy wiersz w pliku to nowa instacja. W ramach każdego wiersza figurują atrybuty oddzielone spacjami, z których pierwszy to atrybut decyzyjny, a kolejne to szereg wartości składające się w szereg czasowy dla tej instancji. Tak jak wspomniano wcześniej, zapis ten pomija dokładny czas zebrania pomiaru. Klasy odpowiadające za wczytywanie danych to odpowiednio `PatientDataImporter2` oraz `DataImporterEamonn`.

4.1.4 Eksportowanie pliku arff

Jedną z funkcji programu jest wygenerowanie pliku `arff` w którym zapisane są już przetworzone dane o instacjach, gotowe do klasyfikacji klasyfikatorami `weki`. Aby stworzyć taki eksporter na własny użytek można posiłkować się klasą abstrakcyjną `ArffExporterBase` i rozszerzyć ją. Eksporter taki zostanie poniżej opisany na przykładzie `RegressionArffBuilder`. Jego kod można znaleźć na listingu 4.2. Klasa nadrzędna wymaga od programisty zaimplementowania dwóch metod oraz kontruktora.

Metoda `setAttributes` służy do tego aby nadać i nazwać atrybuty dla formatu danych jaki zdecydowano użyć w docelowym pliku `arff`. W pliku dla eksperymentu Regression tworzy się tyle par `slope-intercept` (współczynników a i b regresji) na ile części został podzielony sygnał. Należy również pamiętać o tym aby w liście atrybutów zawrzeć również atrybut decyzyjny.

W funkcji buildInstances programista musi obsłużyć przeniesienie danych ze struktury w jakiej trzymał dane po ewentualnych obliczeniach w metodzie `processData` klasy `WorkflowBase` do obiektu `Instances`. Dzięki temu umożliwione będzie wygodne eksportowanie pliku `arff` oraz przeprowadzanie eksperymentu za pomocą klasy `Evaluation`. Ostatnią rzeczą o którą trzeba zadbać, to wywołanie metod `setDestinationClasses` oraz `setAttributes` w kontruktorze tak jak pokazano na wspomnianym listingu.

Listing 4.2: Kod eksportera dla eksperymentu Regression

```
public class RegressionArffBuilder extends ArffExporterBase {

    private List<RegressionRow> input;

    public RegressionArffBuilder(List<RegressionRow> input) {
        this.input = input;
        setDestinationClasses(input);
        setAttributes();
    }

    protected void setAttributes() {
        attrInfo = new FastVector();

        int regressionCount = input.stream().findFirst().get()
                .getRegressionResults().size();

        for (int i = 0; i < regressionCount; i++) {
            attrInfo.addElement(new Attribute(String.format("slope%d", i + 1)));
            attrInfo.addElement(new Attribute(String.format("intercept%d",
                    i + 1)));
        }

        Attribute destClassAttribute = null;
        try {
            destClassAttribute = constructDestinationClassesNominalAttribute(destClasses);
        } catch (Exception e) {
            e.printStackTrace();
        }
        attrInfo.addElement(destClassAttribute);
    }

    @Override
    public Instances buildInstances() {
        instances = new Instances("Regression", attrInfo, input.size());
        instances.setClassIndex(instances.numAttributes() - 1);

        for (RegressionRow row : input) {
            Instance instance = new Instance(attrInfo.size());
            int attrIdx = 0;
            for (RegressionResult regResult : row.getRegressionResults()) {
                instance.setValue(attrIdx++, regResult.getSlope());
                instance.setValue(attrIdx++, regResult.getIntercept());
            }

            int destClassIndex = getIndexOfDestinationClass(row
                    .getDestinationClass());
            instance.setValue(attrIdx, destClassIndex);
            instances.add(instance);
        }
```

```
50          return instances;
51      }
52  }
```

4.2 Serwisy zewnętrzne

4.2.1 Git

Dla zapewnienia jak najwyższej jakości kodu oraz niezawodnego dostępu do niego, używano systemu kontroli wersji, który był dostarczany przez oprogramowanie `git`. Dzięki darmowym serwisom udostępniającym swoje serwery na potrzeby wersjonowania źródeł, takim jak `gitlab.com` zapewnione było bezpieczeństwo danych, a także wspólny dostęp do postępów prac zarówno przez autora, jak i promotora. Na potrzeby obsługi tego narzędzia użyto oprogramowania `SourceTree`, które wyraźnie ułatwia i przyspiesza pracę z systemem `git`.

4.2.2 Travis

Portal `travis.com` jest to serwis Continuous Integration ściśle współpracujący z opisanym wyżej `gitlab.com`. Jego zadaniem jest sprawdzanie każdej kolejnej udostępnionej wersji kodu na serwerze. W momencie gdy pojawia się jego nowa wersja, Travis kompiluje świeżo pobrany kod, uruchamia wszystkie zamieszczone w kodzie testy i ocenia tzw. 'build' pozytywnie lub negatywnie. Dzięki organizacji pracy z takim serwsisem Continuous Integration i pokryciu kluczowych obliczeniowych metod testami, wyłapano wiele błędów implementacyjnych we wczesnym ich etapie.

Rozdział 5

Wnioski

5.1 Analiza wyników

Gdy udało się już uruchomić eksperymenty zgodnie z zamierzeniami, nadszedł czas na interpretację wyników.

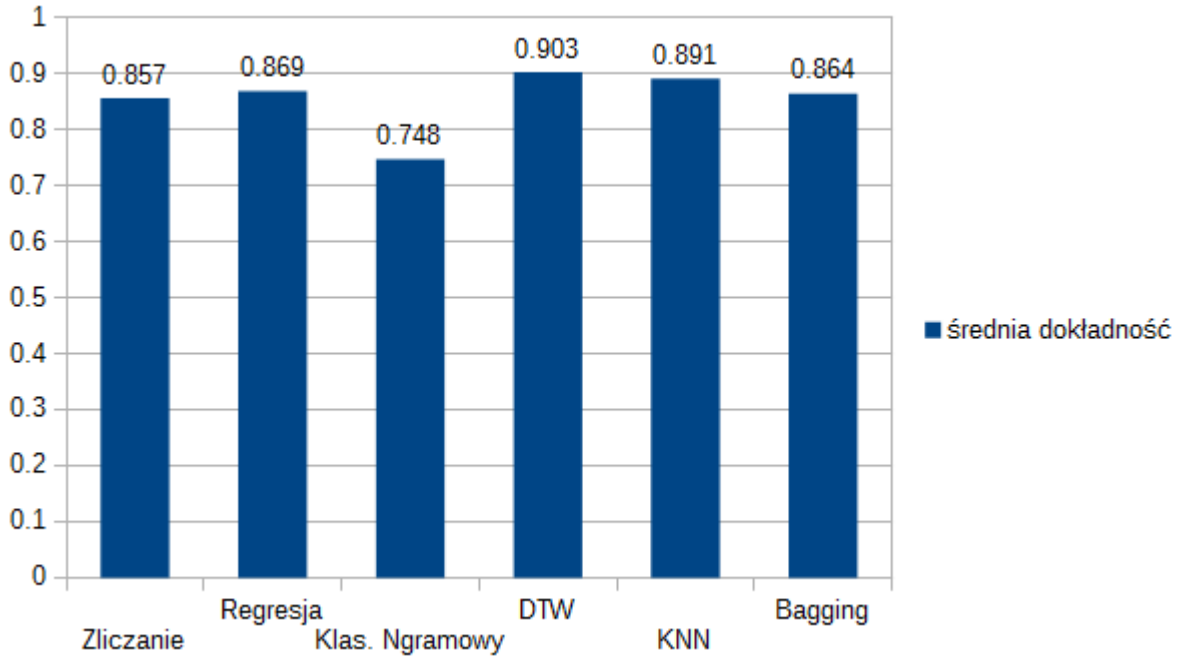
Spośród wszystkich rezultatów jakie zostały sporządzone dla każdego eksperymentu oraz dla wszystkich opcjonalnych zestawów parametrów, wybrano najlepszy zestaw dla każdego z nich. Celem było wybranie obiektywnie najlepszego radzącego sobie dobrze na każdym ze zbiorów danych, aby móc następnie przeprowadzić analizę porównawczą pomiędzy eksperymentami. Kryterium wyboru była największa liczba zbiorów, na których dany zestaw był najlepszy. Przy ewentualnym 'remisie' wybierany był ten, który więcej razy był lepszy od drugiego.

Tym sposobem zostały wybrane następujące uruchomienia eksperymentów:

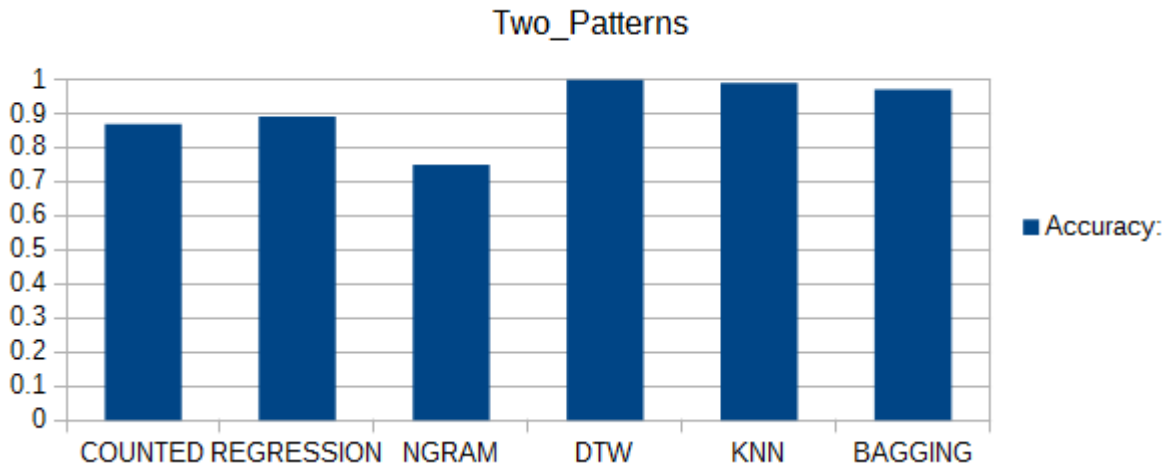
- Zliczanie - części: 5, ngram: 2, alfabet: 11
- Regresja - części: 10
- Klasyfikator ngramowy - ngram: 30, alfabet: 3
- DTW - knn: 7, szerokość okna DTW: 5
- KNN - ngram: 3, alfabet: 11, knn: 3
- Bagging - części: 5, ngram: 3, alfabet: 5

W ramach tych wykonań ekserymentów obliczono średnią dokładność na wszystkich zbiorach, która prezentuje się jak przedstawiono na wykresie 5.1 Zaobserwowano, że eksperymentem cechującym się najlepszą dokładnością na wszystkich z badanych zbiorów jest eksperyment DTW. Jednak jak już wspomniano wcześniej, DTW pojawia się tu jako pewne odniesienie, a nie równy przeciwnik, ponieważ czas przetwarzania tego eksperymentu jest nieporównywalnie większy od konkurentów. Choć za czas spędzony na czekaniu na wyniki można otrzymać stosowne wynagrodzenie - jak można zauważyć na zbiorze TwoPatterns eksperyment ten osiągnął stuprocentową dokładność klasyfikując poprawnie aż 5000 instancji. Rezultaty tego i innych eksperymentów na tym zbiorze można porównać na wykresie 5.2

Miłym zaskoczeniem jest wysoki wynik eksperymentu KNN, ponieważ jego działanie bezpośrednio zależy od algorytmu SAX, w którym pokładano pewne nadzieje. Niestety



Rysunek 5.1: Wykres średniej dokładności eksperymentów



Rysunek 5.2: Dokładność eksperymentów na zbiorze TwoPatterns

zawiódł autorski klasyfikator ngramowy, lecz jego koncept działania nie był mocno wyszukany, więc jego niska dokładność nie jest rozczarowaniem.

Na osobny komentarz zasługuje zbiór Pacjentów, który jak widać na przestrzeni innych zbiorów danych, nie był zbiorem łatwym - zebrał zdecydowanie najsłabsze wyniki na wszystkich klasyfikatorach, co może wskazywać na to, że niekoniecznie podejście do rozwiązania było złe, lecz zbiór był zbyt trudny do rozwikłania.

Z drugiej strony jednak prosta idea nie jest przeszkodą do zdobywania dobrych wyników - czego potwierdzeniem jest znakomity wynik regresji - tutaj wprawdzie nieco zmienionej, bo była ona liczona wielokrotnie dla kolejnych części sygnału w ramach każdej instacji - jednak nadal u podstaw leży nieskomplikowana estymacja danych do funckcji liniowej.

Powstaje pytanie dlaczego skupiono się tylko na mierze dokładności, skoro w większości przypadków badano aż sześć miar. Otóż jak się okazuje, wyniki wszystkich miar w większości eksperymentów (szczególnie w tych dwuklasowych) są do siebie bardzo zbliżone. Obliczono, że odchylenie standardowe miar eksperymentów na zbiorach ECG200, ECGFiveDays, TwoLeadEcg, Yoga, MoteStrain, ItalyPowerDemand, Wafer, oraz Patients jest mniejsze od 0.05. Pozostałe zbiory (ChlorineConcentration, TwoPatterns, InlineSkate) to zbiory wieloklasowe na których zróżnicowanie miar było bardziej wyraźne. Miary F1 oraz PCC często są aż o 0.1 mniejsze niż dokładność co może wskazywać na to, że wiele z instancji nie zostało sklasyfikowanych (Jurek: dobrze to jest? czy bzdura?).

Podsumowując wyniki są zadowalające dzięki eksperymentowi KNN, który nieznacznie odstępuje kroku w jakości wobec matematycznie skomplikowanego DTW, a zdecydowanie deklasuje go w kontekście czasu przetwarzania. Największą zagadką był klasyfikator ngramowy, którego model działania powstał na bazie pomysłu autora, jednak nie okazał się miłą niespodzianką - jedynie potwierdził, że stworzenie dobrego klasyfikatora nie jest prostym zadaniem.

Pełne zestawienie wyników zostało zamieszczone w Dodatku B.

Dodatek A

Podręcznik użytkownika

Dla przejrzystości tego rozdziału przyjęto, że materiały z płyty zostały rozpakowane do ścieżki `D:/TimeSeries/`.

A.1 Uruchomienie projektu w środowisku Eclipse

Projekt programu rozwijany był na środowisku deweloperskim Eclipse Mars, wersja 'Java Developers', na systemie Windows. Projekt zbudowany jest według koncepcji narzędzia Maven, dlatego przy wgrywaniu go do Eclipse należy użyć opcji `File->Import->Existing Maven Project` . Po tym kroku projekt powinien być w stanie bezbłędnie się uruchomić.

A.2 Uruchomienie programu z linii komend

Wraz z kodem źródłowym projektu został dostarczony również wykonywalny plik `.jar` pozwalający na uruchomienie programu bez potrzeby instalacji środowiska deweloperskiego i kompilowania kodu. By przeprowadzić ten proces, należy upewnić się, że dane, które planuje się przetworzyć zlokalizowane są w folderze `/data` w stosunku do pliku wykonywalnego. Nie należy zmieniać drzewa folderów w tej ścieżce.

Przed uruchomieniem programu trzeba sprecyzować jaki wariant eksperymentu zamierza się uruchomić (właściwość `variant`). W przypadku importowania istniejącego pliku arff należy podać dokładną ścieżkę do pliku we właściowości `inputArffPath`. Następnie należy wybrać zbiór danych na jakim eksperyment będzie działał - odpowiada za to właściwość `targetDataset`. Ostatnim parametrem jaki należy doprecyzować w pliku konfiguracyjnym jest rodzaj eksperymentu jaki się uruchomi, sterowanym właściwością `experimentId`. Przykładowy pełny plik konfiguracyjny z podanymi możliwymi wartościami na poszczególnych właściwosciach w komentarzach przedstawiono na listingu A.1. Po dostosowaniu parametrów do potrzeb, można przejść do uruchomienia programu. Potrzebna będzie do tego linia komend, w której należy przejść do ścieżki bazowej `D:/TimeSeries`. Z tego miejsca należy wydać polecenie `java -jar timeseries.jar` i oczekiwać na wyniki. Dla wygody można przekierować wynik przetwarzania do pliku dodając do ubiegłego polecenia `>output.txt`.

Listing A.1: config.properties - plik konfiguracyjny

```
#paths
dataFolderPath = data/
glaucomaDataSet = dataset glaucoma/

# --- Experiment variables ---

# 1, 3, 5 - for regression additionally: 10,20,48
divisionPartsAmount = 3
#2, 3
ngramSize = 3
#3, 5, 11
saxAlphabeatSize = 5

#3, 5, 7
k = 7

dtwSearchRadius = 3

#---------------------------

# --- Target job variant
# 0:ARFFEXPORT 1:ARFFPROCESS 2:FULL
variant = 2

# --- Arff path - file which will be used to run experiment on
# (needed only if variant=1 was picked)
inputArffPath = output/arffOutput/NGRAM/NGRAM for ECGFIVEDAYS with ngram-3 alpha-4.arff

# --- Target dataset - the one which experiment would be launched with
# 0:ECG200 1:ECGFIVEDAYS 2:SAMPLEUNITTEST 3:TWOLEADECG 4:YOGA 5:MOTESTRAIN
# 6:ITALYPOWERDEMAND 7:CHLORINECONCENTRATION 8:TWOPATTERNS 9:WAFER
# 10:INLINESKATE 11:PATIENTS
# -1: run experiment for all datasets (except sample unit test one)
targetDataset = -1

# --- Experiments
# 0:REGRESSION 1:DOMINANT 2:COUNTED 3:NGRAM 4:KNN 5:DTW 6:BAGGING
experimentId = 6

# --- 'constants'
saxOutputLength = 96
crossValidationRepetitions =3
crossValidationFolds = 5
```

Dodatek B

Wyniki

B.1 Wyniki dla najlepszych zestawów parametrów w ramach eksperymentu

B.1.1 Eksperymenty Zliczania, Regresji oraz Ngram

	COUNTED	**REGRESSION**	**NGRAM**
	parts: 5 ngram: 2 alpha: 11	parts: 10	ngram: 30, alpha: 3
Ecg200			
Accuracy:	0.8	0.831667	0.788333
F1 score:	0.798164	0.832295	0.806472
Sensitivity:	0.8	0.831667	0.788333
Specificity:	0.726428	0.784351	0.831649
G-Mean:	0.762291	0.807543	0.809701
PCC:	0.8	0.831667	0.788333
EcgFiveDays			
Accuracy:	0.940422	0.947964	0.837858
F1 score:	0.940479	0.947991	0.839159
Sensitivity:	0.940422	0.947964	0.837858
Specificity:	0.940422	0.947964	0.837858
G-Mean:	0.940422	0.947964	0.837858
PCC:	0.940422	0.947964	0.837858
TwoLeadEcg			
Accuracy:	0.919679	0.965863	0.654045
F1 score:	0.919715	0.965893	0.657586
Sensitivity:	0.919679	0.965863	0.654045
Specificity:	0.919679	0.965863	0.654045

	COUNTED	REGRESSION	NGRAM
G-Mean:	0.919679	0.965863	0.654045
PCC:	0.919679	0.965863	0.654045
Yoga			
Accuracy:	0.825556	0.866465	0.66697
F1 score:	0.825509	0.86646	0.677264
Sensitivity:	0.825556	0.866465	0.66697
Specificity:	0.822888	0.864513	0.63464
G-Mean:	0.824221	0.865488	0.650604
PCC:	0.825556	0.866465	0.66697
MoteStrain			
Accuracy:	0.911688	0.908805	0.802673
F1 score:	0.911702	0.90896	0.807601
Sensitivity:	0.911688	0.908805	0.802673
Specificity:	0.910592	0.909097	0.784759
G-Mean:	0.91114	0.908951	0.793665
PCC:	0.911688	0.908805	0.802673
ItalyPowerDemand			
Accuracy:	0.935219	0.956204	0.809915
F1 score:	0.935256	0.956214	0.810781
Sensitivity:	0.935219	0.956204	0.809915
Specificity:	0.935195	0.956211	0.809777
G-Mean:	0.935207	0.956208	0.809846
PCC:	0.935219	0.956204	0.809915
ChlorineConcentration			
Accuracy:	0.826845	0.88004	0.650698
F1 score:	0.739366	0.819871	0.51047
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	0.740268	0.82006	0.476047
Two patterns			
Accuracy:	0.869533	0.892267	0.7497
F1 score:	0.738998	0.784576	0.485512
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	0.739067	0.784533	0.4994
Wafer			

	COUNTED	REGRESSION	NGRAM
Accuracy:	0.988089	0.995021	0.881863
F1 score:	0.988027	0.99503	0.848667
Sensitivity:	0.988089	0.995021	0.881863
Specificity:	0.928058	0.980909	0.130399
G-Mean:	0.957603	0.987939	0.339108
PCC:	0.988089	0.995021	0.881863
Inlineskate			
Accuracy:	0.825495	0.828425	0.793114
F1 score:	0.390692	0.399968	0.294736
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	0.389231	0.399487	0.275897
Pacjenci			
Accuracy:	0.58046	0.491379	0.58908
F1 score:	0.578826	0.494797	0.591326
Sensitivity:	0.58046	0.491379	0.58908
Specificity:	0.559611	0.491124	0.586083
G-Mean:	0.569931	0.491162	0.587569
PCC:	0.58046	0.491379	0.58908

B.1.2 Eksperymenty DTW, KNN oraz Bagging

	DTW	KNN	BAGGING
	Knn: 7 DtwWin: 5	ngram: 3, alpha: 11, knn: 3,	parts: 5, ngram: 3, alpha: 5,
Ecg200			
Accuracy:	0.833333	0.895	0.803333
F1 score:	0.833177	0.895075	0.801476
Sensitivity:	0.833333	0.895	0.803333
Specificity:	0.733345	0.836007	0.713294
G-Mean:	0.781733	0.864989	0.756963
PCC:	0.833333	0.895	0.803333
EcgFiveDays			
Accuracy:	0.977753	0.99095	0.970588
F1 score:	0.977922	0.990986	0.970603
Sensitivity:	0.977753	0.99095	0.970588
Specificity:	0.977753	0.99095	0.970588

	DTW	**KNN**	**BAGGING**
G-Mean:	0.977753	0.99095	0.970588
PCC:	0.977753	0.99095	0.970588
TwoLeadEcg			
Accuracy:	0.997992	0.968445	0.911933
F1 score:	0.997996	0.968446	0.911954
Sensitivity:	0.997992	0.968445	0.911933
Specificity:	0.997992	0.968445	0.911933
G-Mean:	0.997992	0.968445	0.911933
PCC:	0.997992	0.968445	0.911933
Yoga			
Accuracy:	0.912929	0.924141	0.853333
F1 score:	0.913188	0.924337	0.853572
Sensitivity:	0.912929	0.924141	0.853333
Specificity:	0.908872	0.920632	0.847904
G-Mean:	0.910898	0.922385	0.850614
PCC:	0.912929	0.924141	0.853333
MoteStrain			
Accuracy:	0.964885	0.956761	0.934224
F1 score:	0.96496	0.956786	0.934682
Sensitivity:	0.964885	0.956761	0.934224
Specificity:	0.962841	0.95531	0.936567
G-Mean:	0.963862	0.956035	0.935395
PCC:	0.964885	0.956761	0.934224
ItalyPowerDemand			
Accuracy:	0.958029	0.965328	0.941606
F1 score:	0.958245	0.965334	0.941619
Sensitivity:	0.958029	0.965328	0.941606
Specificity:	0.958084	0.965326	0.941592
G-Mean:	0.958057	0.965327	0.941599
PCC:	0.958029	0.965328	0.941606
ChlorineConcentration			
Accuracy:	0.880143	0.854191	0.764569
F1 score:	0.820249	0.78145	0.656575
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	0.820215	0.781286	0.646854
Two patterns			

	DTW	**KNN**	**BAGGING**
Accuracy:	1	0.989933	0.971367
F1 score:	1	0.979885	0.943337
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	1	0.979867	0.942733
Wafer			
Accuracy:	0.992323	0.998325	0.983482
F1 score:	0.992298	0.998334	0.983363
Sensitivity:	0.992323	0.998325	0.983482
Specificity:	0.955538	0.998259	0.896679
G-Mean:	0.973756	0.998292	0.939078
PCC:	0.992323	0.998325	0.983482
Inlineskate			
Accuracy:	0.856264	0.834725	0.839853
F1 score:	0.503696	0.434377	0.440057
Sensitivity:	0	0	0
Specificity:	0	0	0
G-Mean:	0	0	0
PCC:	0.496923	0.421538	0.439487
Pacjenci			
Accuracy:	0.554598	0.422414	0.534483
F1 score:	0.553333	0.424105	0.525399
Sensitivity:	0.554598	0.422414	0.534483
Specificity:	0.535096	0.411674	0.475673
G-Mean:	0.544758	0.417008	0.504145
PCC:	0.554598	0.422414	0.534483

Bibliografia

[1] Opis przeprowadzenia algorytmu paa w bibliotece jmotif. http://jmotif.github.io/sax-vsm_site/morea/algorithm/PAA.html. Ostatni dostęp: 2015-09-26.

[2] Wzór na odchylenie standardowe. https://en.wikipedia.org/wiki/Standard_deviation#Rapid_calculation_methods. Ostatni dostęp: 2015-09-26.

[3] Szeląg Marcin Błaszczyński Jerzy, Słowiński Roman. Sequential covering rule induction algorithm for variable consistency rough set approaches. pages 987–1002, 2011.

[4] Quek Francis Bryll Robert, Gutierrez-Osuna Ricardo. Attribute bagging:improving accuracy of classifier ensembles by using random feature subsets.

[5] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/, ostatni dostęp 24.09.2015.

[6] Diane J. Cook and Narayanan C. Krishnan. *Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data (Wiley Series on Parallel and Distributed Computing)*. Wiley, 2015.

[7] Roger Jang. *Data Clustering and Pattern Recognition*. http://mirlab.org/jang/books/dcpr/, ostatni dostęp 24.09.2015.

[8] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. pages 2–11, 2003.

[9] Robert C. Martin. Design principles and design patterns. = http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.8854.

[10] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 2010.